Towards Optimally Weighted Physics-Informed Neural Networks in Ocean Modelling

Abstract

Understanding the ocean has particular relevance with the emergence of the climate change phenomenon. Nowadays, this is an essential task, but also very expensive in the computational sense. This work explores the benefits of using physics-informed neural networks (PINNs) for solving partial differential equations (PDEs) related to ocean modeling; such as the Burgers, wave, and advection-diffusion equations. We explore the trade-offs of using data vs. physical models in PINNs for solving partial differential equations. PINNs account for the deviation from physical laws in order to improve learning and generalization. We observed how the relative weight between the data and physical model in the loss function influences training results. Additionally, we compare the variance of our results and analyze the implications of activation functions for training neural network derivatives.

1 Introduction

The ocean is mostly a fluid in almost permanent movement – subjected to currents, tides, waves and other interactions – where different organisms, atmosphere, and continents interact. Therefore, in order to understand the ocean it is necessary to take into account these fundamental characteristics.Understanding the ocean gains particular relevance with the emergence of the climate change phenomenon and, therefore, we need trying to model, forecast and device policies to attempt to mitigate its effects.

Modeling oceanic fluid dynamics is essential but also very expensive in the computational sense. Mathematical models like the Navier-Stokes equations [Temam, 2001] can express some processes of interest in fluid dynamics. Moreover, there are a several equations that are of particular interest when modelling the ocean that are the focus of this work. We refer to [Griffies and Adcroft, 2008] for a brief review on the formulation of models describing the thermo-hydrodynamics of the ocean.

For solving a PDE computationally, methods like finite differences, finite elements and finite volumes, have been successfully implemented in the last decades [Allaire, 2007]. However, when the complexity of the problem is higher, such as in higher dimensional settings, those methods seem to be not efficient enough. Hence, in order to have high performance computing, some methods have been developed, such as reduction of dimensionality for parametric PDEs [Haasdonk, 2017]. Those methods are currently in the frontier of knowledge and have big open questions for certain PDEs, and the state-of-the-art is strongly suggesting the development of deep learning strategies [Ohlberger and Rave, 2015].

Physics-informed neural networks (PINNs) [Raissi *et al.*, 2019] is a technique based on the fact that every smooth (by parts) function can be approximated by a neural network. This fact is complemented by the training of the weights and biases taking into account the information of the PDE and the data, which is can be of two types: for simulations, it is enough to consider a few amount of data, usually in the boundary and in the initial condition; while for inverse problems, we need to take the enough data which depends on the equation and the known results about inverse problems in PDEs, usually a naive requirement is a big amount of data taken from the whole spatio-temporal domain. Other setting for simulations has also been considered; see, e.g., [Sirignano and Spiliopoulos, 2018], where data for prediction can be taken in random interior points. This idea is related to [Mishra and Molinaro, 2022], where the authors show how PINNs are able to predict the solution of a PDE by knowing it in a small part of the domain.

In each general setting for PINNs, these networks are trained by loss terms, one accounting for the deviation from the data and the other from the physics under which the solution is subjected. This extra constraint forces the network to adhere to the physical law underlying the problem, and allows it to improve its performance and to generalize better to unseen data. The physical laws that we investigate consist of temporal and spatial derivatives, which can be easily calculated using the gradient evaluation technique employed for backpropagation in everyday neural network training.

In addition, the balance between the physical and the data parts is a crucial ingredient in the performance of PINNs, such as reported in some works; see, e.g., [van der Meer *et al.*, 2020; Xiang *et al.*, 2021]. As we will see in Section 2, training the PINNs can be seen as a multi-objective learning task, where we want to minimize the error with the data as well as the error with a physical law. Hence, we have three

problems instead of one: minimizing the error with the data, minimizing the error with the physical law, and the combination of training for both objectives.

This paper deals with the problem of predicting the solution of a PDE in space and time relying on PINNs. The PDEs that are considered have a focus on ocean modelling. Our main goal is provide an attempt at determining how the balance between the PDE and the data can improve the performance of PINNs. By answering this question we aim to find a procedure for efficient configuration of parameters with numerical experiments inferred from the characteristics of the problem and the data available.

The contributions of this papers are:

1. to study the influence of the physics information and the scenarios where using it is more effective,
2. a first step towards the automatic parameterization of PINNs using simple oceanographic models as test case, and
3. we study the effects of other hyperparameters as the length and width of the neural network and activation functions on the optimal weight.

The rest of this paper is organized as follows. In Section 2, we deal with the theoretical aspects of our work, that is, we formulate the PINNs problem we study. In Section 3 we present the equations of interest that we want to address with the PINNs technique. In Section 4, we describe the experimental setting of our numerical experiments. Then, in Section 5 analyze the experimental studies involving PINNs for the estimation of the solution and parameters of the PDEs. Finally, in Section 6 we put forward our conclusions of outline out future work.

2 Physics-informed neural networks (PINNs)

PINNs are neural network models that are trained to obey laws in physics described by partial differential equations (PDEs). They can be used used to solve supervised tasks in which we both minimize the error with respect to the data and to the physics law. We define the loss function as follows

$$\ell = (1-\lambda)\ell_{\text{data}} + \lambda\ell_{\text{physics}} \tag{1}$$

where ℓ_{data} is the loss with respect to the data, ℓ_{physics} is the loss with respect to the PDEs of a physics law, and $\lambda \in [0,1]$ is the relative weight of the physics loss. If f^j is the error of each physical condition, then using the mean-squared error for both losses we obtain

$$\ell = (1-\lambda)\sum_{i=0}^{N_u-1}(y_i-\hat{y}_i)^2 + \lambda\sum_{j}\sum_{i=0}^{N_f-1}(f_i^j)^2\,. \tag{2}$$

Any neural network such as fully connected, recurrent, and convolutional networks can be used, where we make use of automatic gradient calculations as used by back-propagation to compute derivatives of the output variables with respect to the input variables. An example of such a network is shown in Fig. 1, with f representing the PDE rewritten as $f = 0$.

The network is designed with input variables x, y, and t and output variables $\hat{u}$ and $\hat{v}$. The output variables are used directly to calculate the data loss term, while for the physics loss term we differentiate the variables with respect to the input variables as needed for the physics loss function. Observe that the neural network must have input variables that correspond to the physics law's derivative terms. That is, if $\frac{\partial u}{\partial x}$ is part of our physics loss function, then x must be an input variable and u an output variable of our neural network.

The objective is to train both in a supervisory manner with measured or simulated data (i. e. ℓ_{data}), as well as training to minimize the departure from the physics law. For a ML perspective, the ℓ_{physics} term ensures that the neural network generalizes better for unseen data by preventing overfitting. The physics loss encourages that the output variables are not just trained to a local region around the input values of the given data, but that also their first and/or second-degree derivations (depending on the PDEs) match with our physical understanding of the model thus spurring better predictive power outside the region of the training data.

As it is a recent topic, there is not much literature about weighted PINNs. Having optimal weights for the loss functions in PINNs could help us to improve the performance of deep learning solvers for PDEs and so make the computations cheaper than regular PINNs. In [van der Meer *et al.*, 2020], the authors propose optimal weights for linear PDEs under the knowledge of boundary values and the differential equation. They use the structure and properties of the PDEs studied. In [Xiang *et al.*, 2021], the authors propose a self-adaptive loss balanced PINNs (lbPINNs) to solve the incompressible Navier-Stokes equation by an empirical search of the weights. In other works, for example, [van Herten *et al.*, 2020; He and Tartakovsky, 2020], the authors also chose empirically the weights to the loss functions in PINNs for their respective applications.

We also hypothesize that, even in cases where there is a sufficient amount of data, adding a physics-informed loss function can potentially lead to a better interpretability of the trained model. This term can be seen as regularization term that would prompt the network to create representations that are consistent with the current knowledge of the phenomena. This, then should allow in the future to have an easier interpretation, assimilation, validation and acceptance by domain experts.

In this work, we study the influence of the weight on a validation quality metric, in particular the relative error of the neural network solution. In addition, we study the effects of other hyperparameters as the length and width of the neural network and activation functions on the optimal weight.

3 Model equations related to the Ocean

We will study the numerical solution by PINNs of the following three ocean modeling related equations:

- *Burgers equation* [Burgers, 1948]: We consider the following Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}\,, \tag{3}$$

  where ν is a diffusion coefficient. This equation usually appears in the context of fluid mechanics, and more

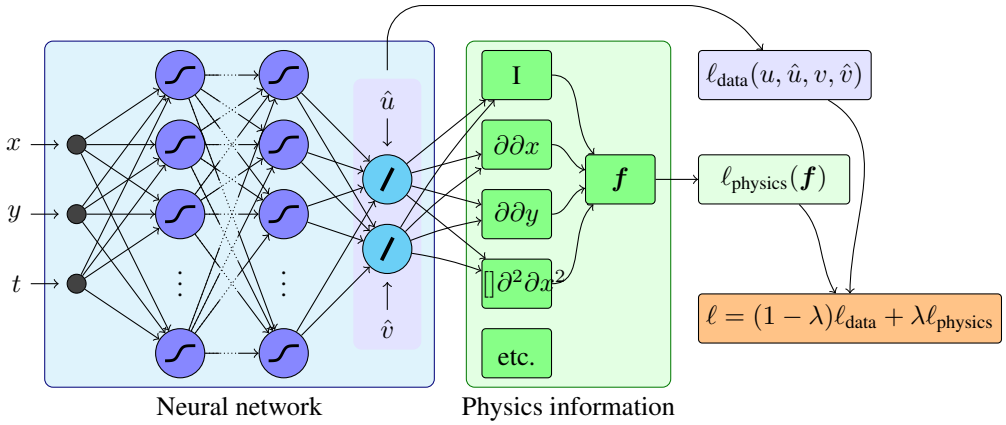


Figure 1: Schematic representation of a physics-informed neural network with inputs x, y, and t; outputs $\hat{u}$ and $\hat{v}$. In this work we applied metaheuristics for determining the optimal hyperparameters for each case. Using automatic gradient calculation we can differentiate the neural network by its input variables and construct a physics error function f. The loss function involves a loss term for the data and a loss term for the physics function.

specifically models one-dimensional internal waves in deep ocean. It represents a hyperbolic conservation law as $\nu \to 0$ and it is the simplest model for analyzing the effect of nonlinear advection and diffusion in a combined way. This equation appears often as a simplification to understand the main properties of the Navier-Stokes equations. It is a one-dimensional equation where the pressure is neglected but the effects of the nonlinear and viscous terms remain, hence as in the Navier-Stokes equations a Reynolds number can be defined [Orlandi, 2000]. Notice that the Burgers equation (3) can be written as a physics loss term of (2) as $f_{\text{Burgers}}[u] = 0$.

- *Wave equation*: under certain suitable conditions, it can be seen as a reduced model of the shallow-water equation which is a well-known model of the water height behavior in coasts and channels [Stoker, 2011], which is a simplification of Navier-Stokes equations in a free boundary under certain suitable an realistic conditions. The wave equation we consider is

$$\frac{\partial^2 \eta}{\partial t^2} - \nabla \cdot (H \nabla \eta) = 0\,, \tag{4}$$

where $\eta : \overline{\Omega} \times \mathbb{R}_0^+ \to \mathbb{R}$ represents the superficial fluctuations of a water container or channel and $H : \overline{\Omega} \to \mathbb{R}_0^+$ is the depth of the water from a reference level. Equation (4) can be written as $f_{\text{waves}}[\eta] = 0$. This is where the physics loss is taken from for this PDE.

- *Advection-diffusion equation*: models the distribution of temperature T in the subsurface of water [Union, 2013]. This equation is obtained by using the energy conservation law and Fourier law of heat conduction. Assuming that there are no sources nor sinks of heat, the advection-diffusion equation takes the form

$$\frac{\partial T}{\partial t} = \nabla \cdot (D \nabla T) - \boldsymbol{u} \cdot \nabla T\,, \tag{5}$$

where $T : \overline{\Omega} \times \mathbb{R}_0^+ \to \mathbb{R}$ is the temperature in K, D is the thermal diffusivity, and $\boldsymbol{u}$ is the groundwater velocity field in $\frac{m}{s}$ satisfying $\nabla \cdot \boldsymbol{u} = 0$, that is, the fluid is assumed to be incompressible. This equation can describe the effects of the fluid flow through the subsurface media on the temperature distribution in the surface. As the previous equations, equation (5) can be written as a physics loss term as $f_{AD}[T] = 0$.

4 Experiments

In this section we describe the datasets preparation and the setting of the PINNs experiments.

4.1 Datasets preparation

In order to generate data for training, validation, and testing, we simulate the PDEs using a Fourier spectral method for the one dimensional model, finite element methods (FEM) for the wave equation and the explicit analytical solution for the advection-diffusion equation. Simulations for the two-dimensional problems were performed using FEniCS [Alnæs *et al.*, 2015].

Burgers equation We simulated the Burgers equation in the spatiotemporal domain $[0, L] \times [0, T]$, using a Fourier spectral method as described in [Basdevant *et al.*, 1986]. The diffusion coefficient is taken as $\nu = \left(\frac{0.01}{\pi}\right)$. We consider the initial condition

$$u(x, 0) = -\sin(\pi x), \text{ with } x \in (-1, 1)\,, \tag{6}$$

and boundary conditions

$$u(1, t) = u(-1, t) = 0, \text{ with } t \in (0, 1)\,, \tag{7}$$

with $N = 512$ spatial points in a uniform grid in $[-1, 1]$ and 100 points in time defined in a uniform grid in $[0, 1]$.

Wave equation We simulate the wave equation in a rectangular domain in the time interval $]0, T_f[$, with Dirichlet boundary conditions for η as

$$\eta = 0, \text{ on } \partial\Omega \times]0, T_f[\,, \tag{8}$$

and initial conditions on Ω

$$\begin{aligned} \eta(x,y,0) &= \exp\left(-10\cdot\left((x-0.5)^2+(y-0.75)^2\right)\right), \\ \frac{\partial\eta}{\partial t}(x,y,0) &= 0\,. \end{aligned} \tag{9}$$

On the other hand, the depth H is given by

$$H(x,y) = (1-x)(2-\sin(3\pi y)) \quad \text{in } \overline{\Omega}\,. \tag{10}$$

We implement FEM in this equation considering Lagrange finite elements of degree 1. The time scheme is explicit and $T_f = 1.0$, $n = 100$.

Advection-diffusion equation We simulate the two-dimensional advection-diffusion equation (5) for $T = T(x,y,t)$ in $\Omega\times(0,T_f)$ where $\Omega = (0,L_x)\times(0,L_y)$, where $L_x = L_y = 1.0$ and $T_f = 1.0$. In addition, we consider a constant diffusion coefficient, $D = 0.02$; the velocity is assumed to have only the horizontal components, that is, $\boldsymbol{u} = (u,v)$, where $u = \cos(\phi)$ and $v = \sin(\phi)$, and $\phi = 22.5°$. Let us consider the function

$$\tilde{T}(x,y,t) = \frac{1}{4t+1}\exp\left(-\frac{x^2+y^2+t^2(u^2+v^2)-2t(xu+yv)}{D(4t+1)}\right). \tag{11}$$

Notice that $T = \tilde{T}$ is a solution for the advection-diffusion equation (5), so we will take the following boundary and initial conditions:

$$T|_{\partial\Omega\times]0,T_f[} = \tilde{T}|_{\partial\Omega\times]0,T_f[},\ T|_{\Omega\times\{t=0\}} = \tilde{T}|_{\Omega\times\{t=0\}}.$$

We generate this solution in a uniform unit square mesh with 40 points in each spatial dimension. This setting follows [He and Tartakovsky, 2020], where the authors study PINNs for advection-difussion equations, also considering different weights.

4.2 Settings for PINNs

In this work we study the behavior of the relative error (validation loss) with respect to the weight in the physical part, for Burgers, wave, and advection-diffusion equations.

As part of the experimentation we assess different activation functions. As our optimization problem is non-convex, the stochastic gradient-descent Adam optimizer was used [Kingma and Ba, 2017] with different learning rates depending on the problem. The weights of the neural network are initialized randomly using Xavier's initialization method [Glorot and Bengio, 2010]. As we intend to analyze how activation functions play a role in learning the derivations of PDEs the $\tanh$, GELU, Softplus, LogSigmoid, Sigmoid, TanhShrink, CELU, Softsign, and ReLU activation functions.

We select a sample subset $(X_i, Y_i)\forall i \in [0, N_u - 1]$ at random from the entire solution space for training the data loss, with $X_i \in \mathbb{R}^D$ the D features and $Y_i \in \mathbb{R}^P$ the labels of dimension P. For the physics loss we select a subset $X_i \forall i \in [0, N_f - 1]$ at random for which we evaluate the PDE error. Note that for the physics loss the solution is not needed, allowing applications to be able to train even when few data are available of the solution. In general we pick $N_u \ll N_f$.

5 Results and discussion

See Table 1 for an overview of the used parameters for each of the models. The hyperparameters, such as the number and size of the hidden layers; the optimizer; and the number of epochs, were chosen either manually or by a grid search. All experiments were executed using Google Colab, a platform to run notebooks on fast graphics processing units (GPUs). The ability of GPUs to run calculations in parallel greatly enhances the speed of evaluating the neural network and calculating the gradients.[1]

To be able to compare the performance between models, the validation data loss is calculated over the entire data set and then normalized as

$$\text{Relative Error} = \frac{\sqrt{\text{MSE}(Y_{val}, \hat{Y}_{val})}}{\|Y_{val}\|_2}, \tag{12}$$

where Y_{val} are the labels of the validation data set, $\hat{Y}_{val}$ the output of the neural network, and MSE the mean squared error. In the case of advection-diffusion the labels Y_i have been normalized to have zero mean and unit standard deviation to bring down the scale of the temperature, which is in Kelvin, to a range around zero. This obviates the need for the division when calculating the relative error. Additionally, to reduce the jitter in the loss while training, we use the lowest validation data loss of the last 250 epochs to calculate the relative error.

Learning the optimal value of λ, the relative weighting of the physics loss with respect to the data loss is a metalearning objective in order to improve subsequent training of the same PINN. In Fig. 2 we show the results of evaluating different problems using different values for N_u to find an optimal value of λ. A sharp rise in the relative error can be observed at $\lambda = 0.0$ and $\lambda = 1.0$, which are equivalent to training using solely the data loss or physics loss terms respectively. We note that the optimal value for λ depends on the problem at hand and specifically on the scale of the data loss versus the physics loss. Normalizing the data set labels would in part solve the problem, but the scale of the physics loss remains hard to normalize. In our experiments however, we note that for typical values of λ the comparison of the data and physics loss remains roughly $0.1 < \ell_{\text{data}}/\ell_{\text{physics}} < 10$.

Per problem we observe that at decreasing values for N_u, the optimal value of λ shifts slightly towards 1.0. This trend suggests that as data become more sparse, the physics loss takes higher importance. As less data are available, the information input from the physics loss helps training significantly, while the reverse holds as more data are available. At the limit where N_u is sufficiently large to be able to train the network autonomously, the optimal value of λ tends to zero.

In cases where few data points are known (i.e. N_u is small), the choice of the sample subset can have a large influence on training results. In Fig. 3a the relative error is shown for 25 random data subset selections with and without the physics loss. The added physics loss term reduces the relative error,

[1] The data sets, source code, and results are available online at https://github.com/Inria-Chile/assessing-pinns-ocean-modelling and are released under the CeCILL license.

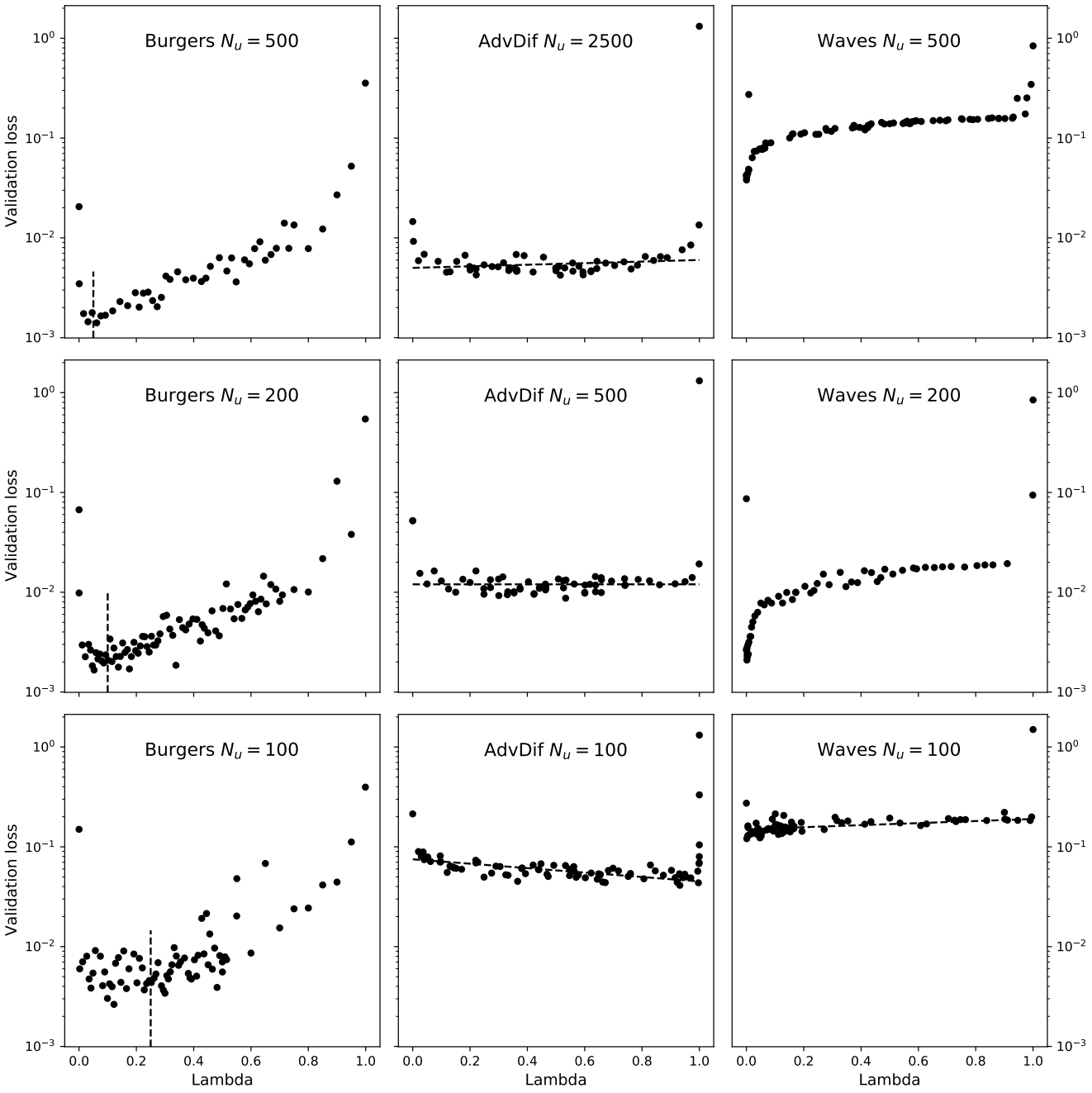
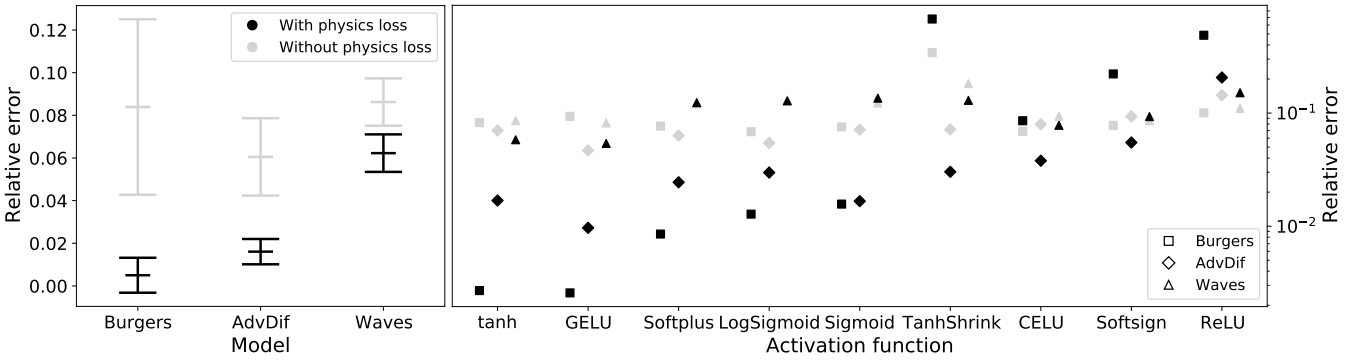


Figure 2: The relative error for different values of N_u and for different problems. Each dot represents the final value of the relative error after training. We observe how the choice of λ affects training results, and we observe a slight shift to higher λ values as N_u decreases. The dashed lines were added manually to aid in observing trends. For Burgers we note that the minimum shifts towards 1.0 as N_u decreases, and for advection-diffusion the slope becomes more negative as N_u decreases. The wave equation is more less evident, but it can be seen that for $N_u = 500$ and $N_u = 200$ the minimum lies around zero, while for $N_u = 100$ the trend has flattened and the results would be similar for $\lambda \in [0.1, 0.9]$ roughly.

Table 1: Parameters for each of the models. Here epochs are the numbers of iterations over the training set, and N_u and N_f are the number of points used for the data loss and the physics loss respectively. N_u^{val} is the number of data points used to for the validation data loss.

Model	NN setup	Optimizer	Epochs	N_u	N_f	N_u^{val}
Burgers	8×20	Adam (lr=0.001)	25000	100, 200, 500	12800	25600
Advec.-Diffusion	3×40	Adam (lr=0.002)	20000	100, 500, 2500	5000	168100
Wave	5×50	Adam (lr=0.001)	10000	100, 200, 500	10000	21800



(a) Errors with/without physics loss.

(b) Relative error using different activation functions.

Figure 3: Summary of the experimental results. Fig. 3a shows results for 25 runs with and without physics loss for each of the models. For Fig. 3b, N_u is 200, 500, and 200 and λ is 0.1, 0.5, and 0.002 for the Burgers, advection-diffusion, and wave equations respectively.

averaged over all three models, by 65% and reduces its standard deviation by 56%. PINNs thus improve training results and reduce variability when the number of data points is low.

An analysis of how activation functions play a role in learning the derivations of PDEs, a comparison is made with nine activation functions in Fig. 3b. Here we compare the $\tanh$, GELU, Softplus, LogSigmoid, Sigmoid, TanhShrink, CELU, Softsign, ReLU activation function for their performance. It can be observed that especially the $\tanh$ and GELU activation functions and to a lesser extend Softplus work well. The ReLU function produces poor results due to its inability to learn second or higher derivatives. Due to the fact that a DNN using the ReLU activation function produces a piece-wise linear function [Liu and Liang, 2021], its second derivative is zero except at the points where the linear segments meet, where it will be a Dirac function at $x = 0$, inhibiting learning second derivatives or higher. The Softsign and CELU functions have similar problems given their discontinuity of the second derivative at $x = 0$.

6 Final remarks

Information from the PDE serves in cases where data availability is limited to the extent that it would either be unable to train the network satisfactorily using solely the data, or where results would vary widely depending on the selection of the data points. The addition of the physics loss term both improves training results and improves the stability of the training, even more so when the weight in the loss function is optimal.

Finding the optimal weight of the loss function is a complex problem and this is, to the best of our knowledge, the first study to understand the behavior of it with respect to different hyperparameters and PDEs. In the future, we expect to find the optimal weights using only the data available. In addition, it is also the first attempt at the study of PINNs for ocean modelling equations with very simplified geometries. We expect to extend our results to more complex and realistic geometries in order to use our results in real applications.

As λ has a dual role: the prior confidence in the training data vs. the PDE and the scaling of these two terms, future work could understand each role of this parameter, which could be a possible explanation of why the optimal weight in some of our experiments are so different between different models. In addition, as we observed for very small data sets, it would be interesting to find what "good" sample subsets of the problem for training are more stable under optimization. This would be the first step to estimate the minimal amount of data required to train the neural networks in a robust way.

Future work on this topic is needed as it is required to develop better models for understanding the ocean and integrate those models with others that explain the interaction between organisms and the environment. This is essential because of the constantly changing nature of the ocean.

We believe that PINNs have the potential of providing such tools, and therefore, serve as a building block of a more comprehensive solution that addresses the ultimate issue of understanding and mitigating the climate change crisis. However, this potential also comes with important risks as using an inadequate, biased, or non-transparent tool could lead instead to a deterioration. In this direction, as we already mentioned

in the paper, we are interested in combining PINNs with explainable AI and powerful visualization tools. Similarly, extended validation is necessary by domain experts.

Ethical Statement

There are no ethical issues.

Acknowledgments

References

[Allaire, 2007] Grégoire Allaire. *Numerical analysis and optimization: an introduction to mathematical modelling and numerical simulation*. Oxford university press, 2007.

[Alnæs *et al.*, 2015] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[Basdevant *et al.*, 1986] Cea Basdevant, M Deville, P Haldenwang, JM Lacroix, J Ouazzani, R Peyret, Paolo Orlandi, and AT Patera. Spectral and finite difference solutions of the burgers equation. *Computers & fluids*, 14(1):23–41, 1986.

[Burgers, 1948] J.M. Burgers. A mathematical model illustrating the theory of turbulence. volume 1 of *Advances in Applied Mechanics*, pages 171–199. Elsevier, 1948.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 5 2010. PMLR.

[Griffies and Adcroft, 2008] Stephen M Griffies and Alistair J Adcroft. Formulating the equations for ocean models. *Ocean modeling in an eddying regime*, 177:281–317, 2008.

[Haasdonk, 2017] Bernard Haasdonk. Reduced basis methods for parametrized pdes–a tutorial introduction for stationary and instationary problems. *Model reduction and approximation: theory and algorithms*, 15:65, 2017.

[He and Tartakovsky, 2020] QiZhi He and Alexandre M Tartakovsky. Physics-informed neural network method for forward and backward advection-dispersion equations. *arXiv preprint arXiv:2012.11658*, 2020.

[Kingma and Ba, 2017] Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization, 2017.

[Liu and Liang, 2021] Bo Liu and Yi Liang. Optimal function approximation with ReLU neural networks. *Neurocomputing*, 435:216–227, 2021.

[Mishra and Molinaro, 2022] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes. *IMA Journal of Numerical Analysis*, 42(2):981–1022, 2022.

[Ohlberger and Rave, 2015] Mario Ohlberger and Stephan Rave. Reduced basis methods: Success, limitations and future challenges. *arXiv preprint arXiv:1511.02021*, 2015.

[Orlandi, 2000] Paolo Orlandi. *The Burgers equation*, pages 40–50. Springer Netherlands, Dordrecht, 2000.

[Raissi *et al.*, 2019] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[Sirignano and Spiliopoulos, 2018] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:339–1364, 12 2018.

[Stoker, 2011] James Johnston Stoker. *Water waves: The mathematical theory with applications*, volume 36. John Wiley & Sons, 2011.

[Temam, 2001] Roger Temam. *Navier-Stokes equations: Theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

[Union, 2013] J Ind Geophys Union. Advection diffusion equation models in near-surface geophysical and environmental sciences. *J. Ind. Geophys. Union (April 2013)*, 17(2):117–127, 2013.

[van der Meer *et al.*, 2020] Remco van der Meer, Cornelis Oosterlee, and Anastasia Borovykh. Optimally weighted loss functions for solving PDEs with neural networks. *arXiv preprint arXiv:2002.06269*, 2020.

[van Herten *et al.*, 2020] Rudolf LM van Herten, Amedeo Chiribiri, Marcel Breeuwer, Mitko Veta, and Cian M Scannell. Physics-informed neural networks for myocardial perfusion mri quantification. *arXiv preprint arXiv:2011.12844*, 2020.

[Xiang *et al.*, 2021] Zixue Xiang, Wei Peng, Xiaohu Zheng, Xiaoyu Zhao, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks for the incompressible navier-stokes equations. *arXiv preprint arXiv:2104.06217*, 2021.